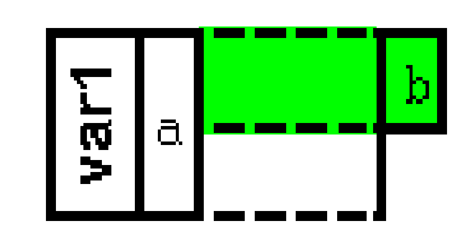


Declaración

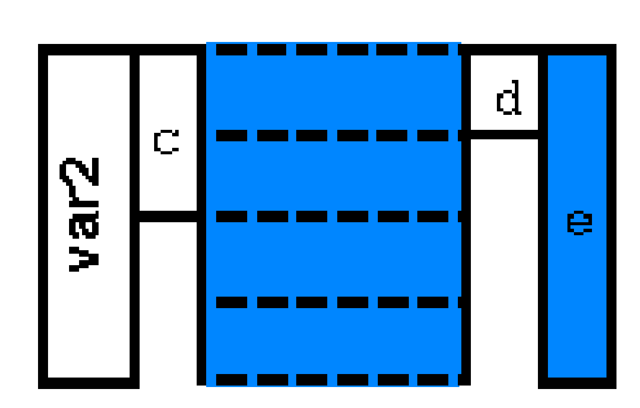
```
union etiqueta {
    tipo campo_1;
    tipo campo_2;
    tipo campo_3;
    . . .
    . . .
    tipo campo_n
} lista_de_variables;
```

Ubicación y tamaño de los campos de una unión en la memoria

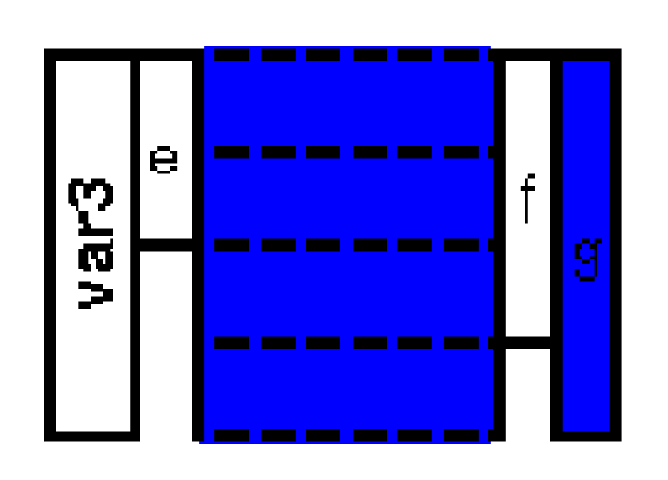
```
union ej1 {int a;
           char b;} var1;
```



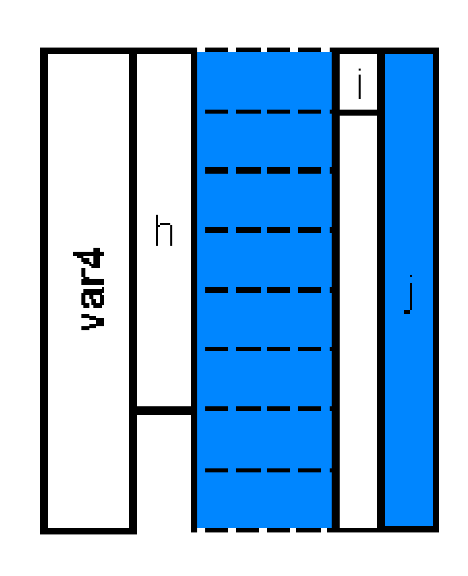

```
union ej2 {int c;
           char d;
           float e;} var2;
```



```
union ej3 {int e;
           char f[3];
           float g;} var3;
```




```
union ej4 {int h[3];
           char i;
           float j[2];} var4;
```

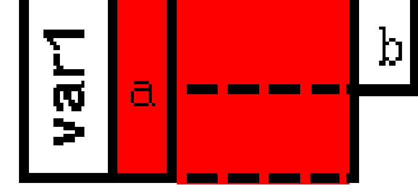


erencia y asignación a campos en una un

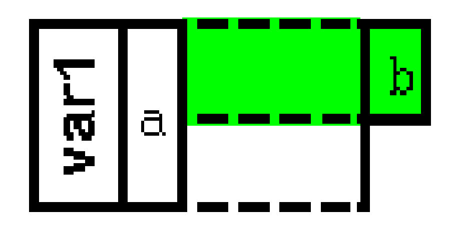
- Referencia a un campo

```
union ej1 {int a;
           char b;} var1;
```

var1.a



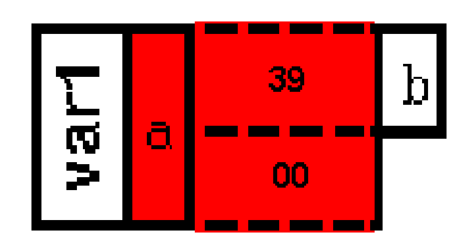
var1.b



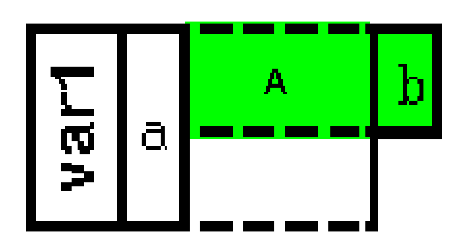
Asignación

```
union ej1 {int a;
           char b;} var1;
```

var1.a=39

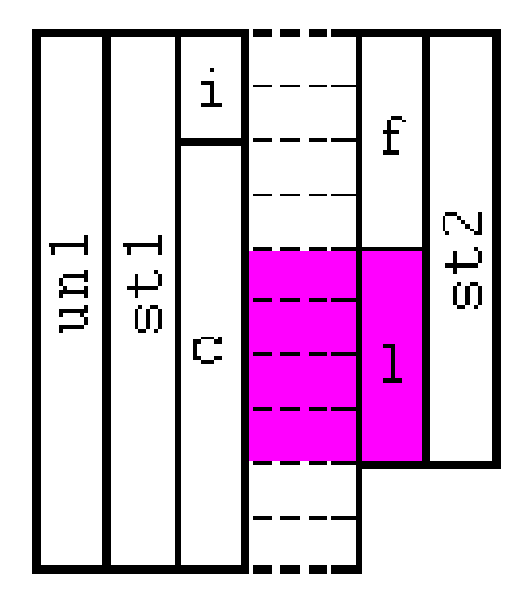


var1.b='A'



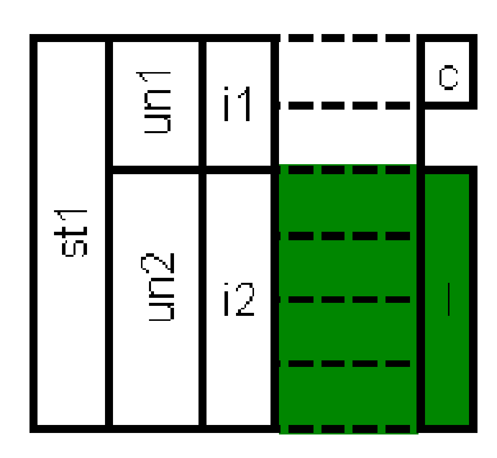
Uniones de estructuras

```
union unst {struct{int i;
                   char c[8];} st1;
            struct{float f;
                   long l;} st2;} un1
```



Estructuras de uniones

```
struct stun {  unión {    int  i1;
                          char c;}un1;
               unión {    int  i2[2];
                          long l;}un2;} st1;
st1;
```



CAMPOS DE BITS

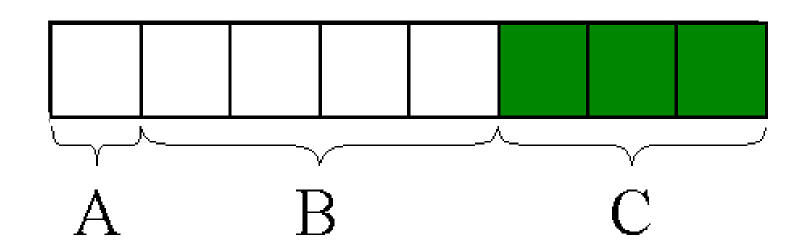
CAMPOS DE BITS

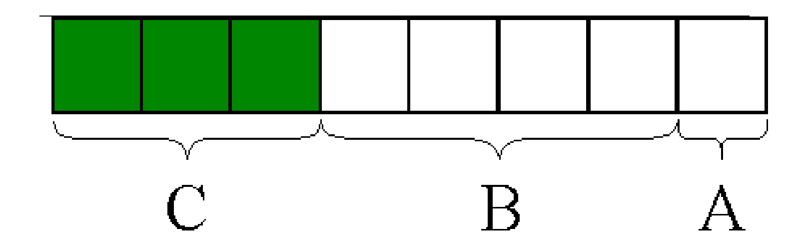
Declaración

```
struct etiqueta {  tipo entero  campo_1: # de bits;
                   tipo entero  campo_2: # de bits;
                   tipo entero  campo_3: # de bits;
                    . . .
                    . . .
                   tipo entero  campo_n: # de bits; } lista_var;
```

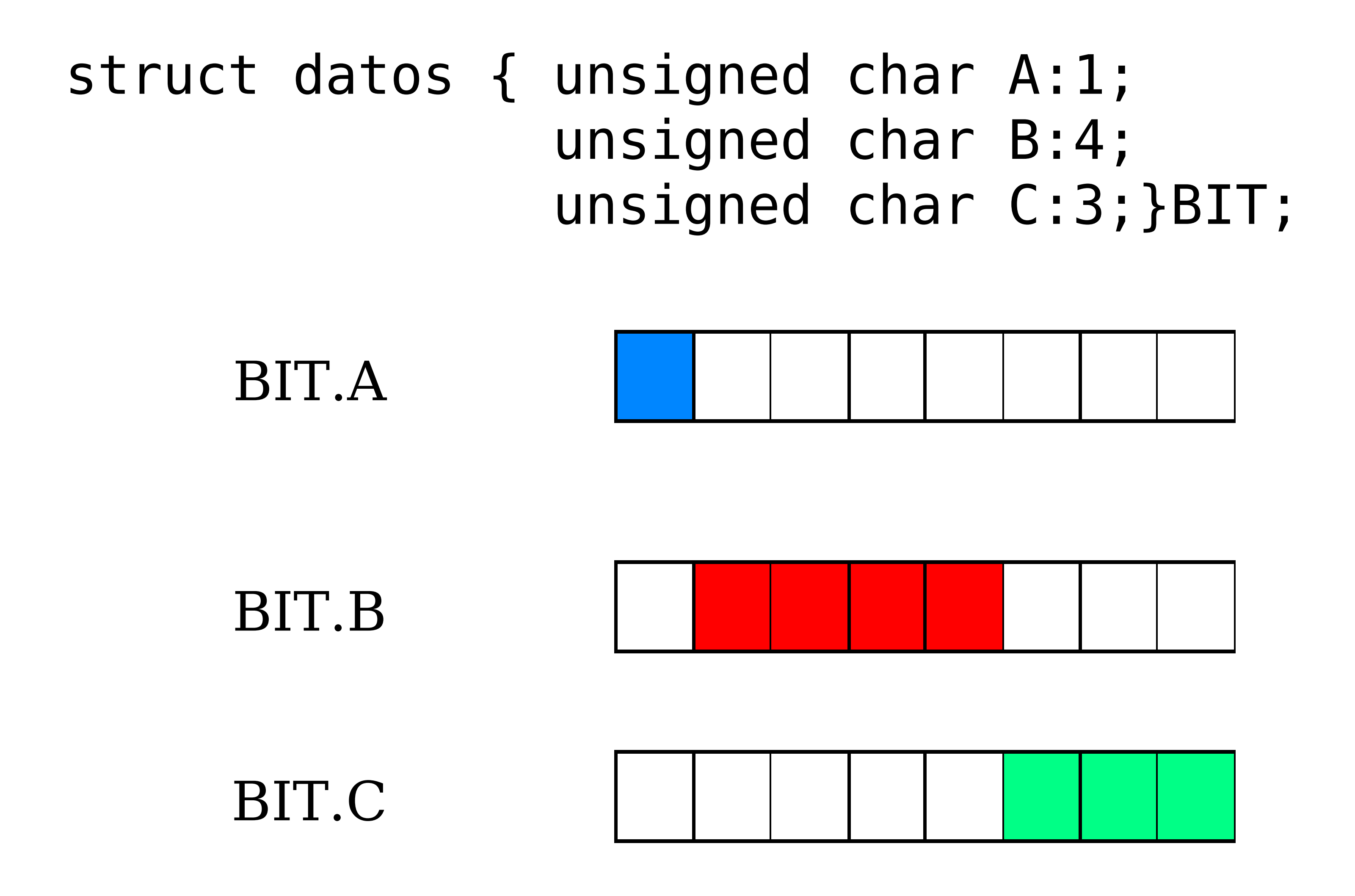
Ubicación de los campos de bits en la memoria

```
struct datos { unsigned char A:1;
               unsigned char B:4;
               unsigned char C:3;};
```





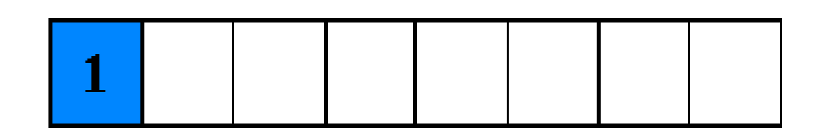
Referencia a un campo de bits



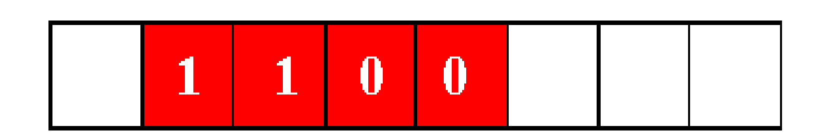
Asignación de valores en un campo de bits

```
struct datos { unsigned char A:1;
               unsigned char B:4;
               unsigned char C:3;}BIT;
```

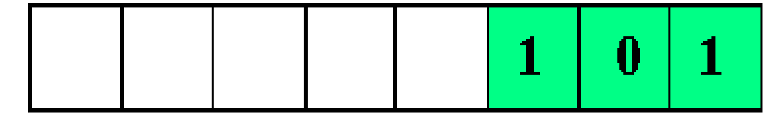
BIT.A=1



BIT.B=0xC



BIT.C=5



Operadores a nivel de bit

AND (&)

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

```
char a,b,c;
```

`a=0xaa;`

1	0	1	0	1	0	1	0

`b=0x55;`

0	1	0	1	0	1	0	1

`c=a&b;`

0	0	0	0	0	0	0	0

OR (|)

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

```
char a,b,c;
```

`a=0xaa;`	1	0	1	0	1	0	1	0
`b=0x55;`	0	1	0	1	0	1	0	1
`c=a|b;`	1	1	1	1	1	1	1	1

XOR (^)

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

```
char a,b,c;
```

```
a=0xaa;
```

1	0	1	0	1	0	1	0

```
b=0x55;
```

0	1	0	1	0	1	0	1

```
c=a&b;
```

1	1	1	1	1	1	1	1

NOT (~)

a	~a
0	1
1	0

```
char a,b;
a=0xaa;
b=~a;
```

1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1

Desplazamiento izquierdo (<<)

- Produce el desplazamiento de los bits de la variable hacia la izquierda en la cantidad expresada
- Inserta ceros en los bits menos significativos
- Su sintaxis es variable1 = variable2 << cantidad_desplaz;

```
char a,b;

a=0xaa;

b=a<<3;
```

1	0	1	0	1	0	1	0

0	1	0	1	0	0	0	0

Desplazamiento derecho (>>)

- Produce el desplazamiento de los bits de la variable hacia la derecha en la cantidad expresada

- Inserta ceros en los bits mas significativos

- Su sintaxis es variable1 = variable2 >> cantidad_desplaz;

```
char a,b;

a=0xaa;

b=a>>3;
```

1	0	1	0	1	0	1	0

0	0	0	1	0	1	0	1

ENUMERACIONES

ENUMERACIONES

Declaración

```
enum etiqueta{constante_1;
              constante_2;
              constante_3;
               . . .
               . . .
              constante _n;} lista_de_variables;
```

Enum color

{ blanco,negro,verde,azul,rojo};

Enum dias

{ **lunes=2**,martes,miercoles,jueves,viernes};

Enum cargos

{ titular,asociado,adjunto,**jtp=10**,atp1,atp2};

Enum mes

{**ene=10**,feb,mar,**abr=20**,may,jun,

jul=30,ago,sep,**oct=40**,nov,dic};